LAB 2 18/03/2022
MIPS Sample Programs

1. MIPS assembly, write an assembly language version of the following C code segment

```
int A[100], B[100];
for (i=1; i < 100; i++) {
  A[i] = A[i-1] + B[i];
}
```

At the beginning of this code segment, the only values in registers are the base address of array A and B in registers $a0 and $a1.Avoid the use of multiplication instructions-they are unnecessary.

The MIPS assembly sequence is as follows

```
 li $t0, 1          # Starting index of i
 li $t5, 100        # Loop bound

loop:
 lw $t1, 0($a1)  # Load A[i-1]
 lw $t2, 4($a2)    # Load B[i]
 add $t3, $t1, $t2  # A[i-1] + B[i]
 sw $t3, 4($a1)  # A[i] = A[i-1] + B[i]
 addi $a1, 4    # Go to i+1
 addi $a2, 4    # Go to i+1
 addi $t0, 1    # Increment index variable
 bne $t0, $t5, loop  # Compare with Loop Bound

halt:
 nop
```

2. Convert the following C statements to equivalent MIPS assembly language. Assume that the variables f, g, I and j are assigened to registers $s0, $s1, $s2 and $s3 respectively. Assume that the base address of the array A and B are in registers $s6 and $s7 respectively.

a) f = g + h + B[4]

```
lw $t0, 16($s7)
add $s0, $s1, $s2
add $s0, $s0, $t0
```

b) f = g – A[B[4]]

```
 lw $t0,16($s7)
sll $t1, $t0, 2
add $t2, $t1, $s6
lw $t3, 0($t2)
sub $s0, $s1, $t3
```

Convert the following program into machine code.

```
0xFC00000C      start: ..................
0xFC000010      loop: addi $t0,$t0,-1
0xFC000014            sw $t0, 4($t2)
0xFC000018            bne $t0, $t3, loop
0xFC00001C            j start
```

`addi $t0,$t0,-1`

-1 = 0xFFFF

0010 00/01 000/0 1000 /1111 1111 1111 1111

8	8	8	-1
Op(6)	rs(5)	rt(5)	immediate(16)

`sw $t0, 4($t2)`

1010 11/01 010/0 1000 /0000 0000 0000 0100

43	10	8	4
Op(6)	rs(5)	rt(5)	immediate(16)

`bne $t0, $t3, loop`

target address = (immediate * 4) + address of the following instruction

immediate = (target address – address of the following instruction) / 4

= (FC000010 – FC00001C) / 4

= - C /4

= -3 → 1111 1111 1111 1101

Or convert to bainary first

= 1111 1100 0000 0000 0000 0000 0001 0000 –
1111 1100 0000 0000 0000 0000 0001 1100

= 1111 1100 0000 0000 0000 0000 0001 0000 +
0000 0011 1111 1111 1111 1111 1110 0100

= 1111 1111 1111 1111 1111 1111 1111 0100 / 4 → srl by 2

= 11 1111 1111 1111 1111 1111 1111 1101

immediate is 16 only so immediate = 1111 1111 1111 1101

0001 01/01 000/0 1011 /1111 1111 1111 1101

5	8	11	-3
Op(6)	rs(5)	rt(5)	immediate(16)

`j start`

target address = last 4 bits of PC : (immediate * 4)

immediate = first 28 bits from target address / 4

= C00000C / 4 = 3000003

= 1100 0000 0000 0000 0000 0000 1100 / 4 → srl by 2

= 11 0000 0000 0000 0000 0000 0011

0000 10/11 0000 0000 0000 0000 0000 0011

2	3000003
Op(6)	immediate (26)

4 factorial Program

```
data
        msg: .asciiz "Enter a number"
        answer: .asciiz "\nFactorial is: "
        .text
        # welcome message
        li $v0, 4
        la $a0, msg
        syscall
        # read integer
        li $v0, 5
        syscall
        # print the integer
        move $a0, $v0
        li $v0, 1
        syscall
        jal calculate_factorial
        move $a1, $v0
        li $v0, 4
        la $a0, answer
        syscall
        move $a0, $a1
        li $v0, 1
        syscall
        li $v0, 10
        syscall
        calculate_factorial:
                addi $sp, $sp-4
                sw $ra, ($sp)
                li $v0, 1
                multiply:
                        beq $a0, $zero, return
                        mul $v0, $v0, $a0
                        addi $a0, $a0, -1
                        j multiply
                return:
                lw $ra, ($sp)
                jr $ra
```